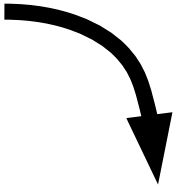
A whirlwind tour of scikit-learn

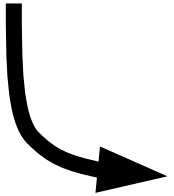
Andreas Mueller

Alfred P. Sloan
FOUNDATION

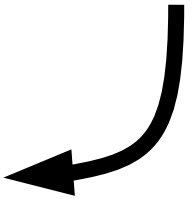

universitätbonn
ais




Amazon
Machine
Learning



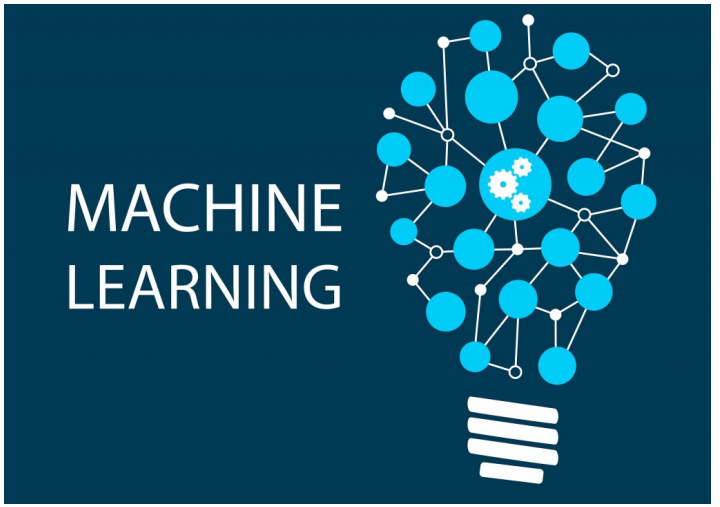

NYU
Center for
Data Science




COLUMBIA UNIVERSITY
Data Science Institute

What is scikit-learn?


python


MACHINE
LEARNING

Classification
Regression
Clustering
Semi-Supervised Learning
Feature Selection
Feature Extraction
Manifold Learning
Dimensionality Reduction
Kernel Approximation
Hyperparameter Optimization
Evaluation Metrics
Out-of-core learning
……

okc

dataiku
Write your own data story

Spotify®

The New York Times

9500+ research papers


Alexandre Gramfort
agramfort


Alexander Fabisch
AlexanderFabisch


Alexandre Passos
alextp


Andreas Mueller
amueller


Arnaud Joly
arjoly


Brian Holt
bdholt1

bthirion
bthirion


Chris Filo Gorgolewski
chrisfilo


David Cournapeau
cournape


Duchesnay
duchesnay


David Warde-Farley
dwf


Fabian Pedregosa
fabianp


Gael Varoquaux
GaelVaroquaux


Gilles Louppe
glouppe


Jake Vanderplas
jakevdp


Jaques Grobler
jaquesgrobler


Jan Hendrik Metzen
jmetzen


Jacob Schreiber
jmschrei


Joel Nothman
jnothman


Kyle Kastner
kastnerkyle


Lars
larsmans


Loïc Estève
lesteve


Shiqiao Du
lucidfrontier45


Mathieu Blondel
mblondel


Manoj Kumar
MechCoder


Noel Dawe
ndawe


Nelle Varoquaux
NelleV


Olivier Grisel
ogrisel


Paolo Losi
paolo-losi


Peter Prettenhofer
pprett


(Venkat) Raghav (Rajagopalan)
raghavrv


Robert Layton
robertlayton


Ron Weiss
ronw


Satrajit Ghosh
satra


sklearn-ci


sklearn-wheels


Tom Dupré la Tour
TomDLT


Vlad Niculae
vene


Virgile Fritsch
VirgileFritsch


Vincent Michel
vmichel

Wei Li
weilinear

Yaroslav Halchenko
yarikoptic

Documentation of scikit-learn 0.19.1

Quick Start

A very short introduction into machine learning problems and how to solve them using scikit-learn. Introduced basic concepts and conventions.

User Guide

The main documentation. This contains an in-depth description of all algorithms and how to apply them.

Other Versions

- Development version
- All available versions
- PDF documentation

Tutorials

Useful tutorials for developing a feel for some of scikit-learn's applications in the machine learning field.

API

The exact API of all functions and classes, as given by the docstrings. The API documents expected types and allowed features for all functions, and all parameters available for the algorithms.

Additional Resources

Talks given, slide-sets and other information relevant to scikit-learn.

Development

Information on how to contribute. This also contains useful information for advanced users, for example how to build their own estimators.

Flow Chart

A graphical overview of basic areas of machine learning, and guidance which kind of algorithms to use in a given situation.

FAQ

Frequently asked questions about the project and contributing.

Related packages

Other machine learning packages for Python and related projects. Also algorithms that are slightly out of scope or not well established enough for scikit-learn.

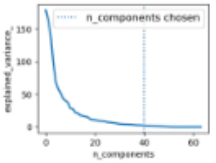
http://scikit-learn.org/

Examples

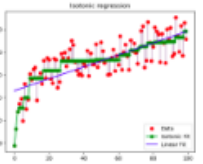
General examples

General-purpose and introductory examples for scikit-learn.

Concatenating multiple feature extraction methods

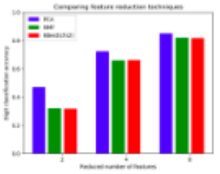


Pipelining: chaining a PCA and a logistic regression

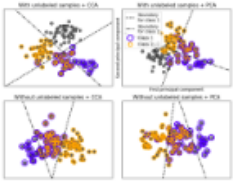


Isotonic Regression

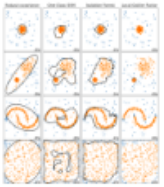
Face completion with a multi-output estimators



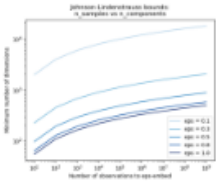
Selecting dimensionality reduction with Pipeline and GridSearchCV



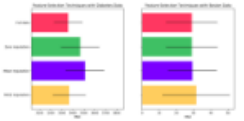
Multilabel classification



Comparing anomaly detection algorithms for outlier detection on toy datasets



The Johnson-Lindenstrauss bound for embedding with random projections



Imputing missing values before building an estimator

Basic API

Representing Data

one sample

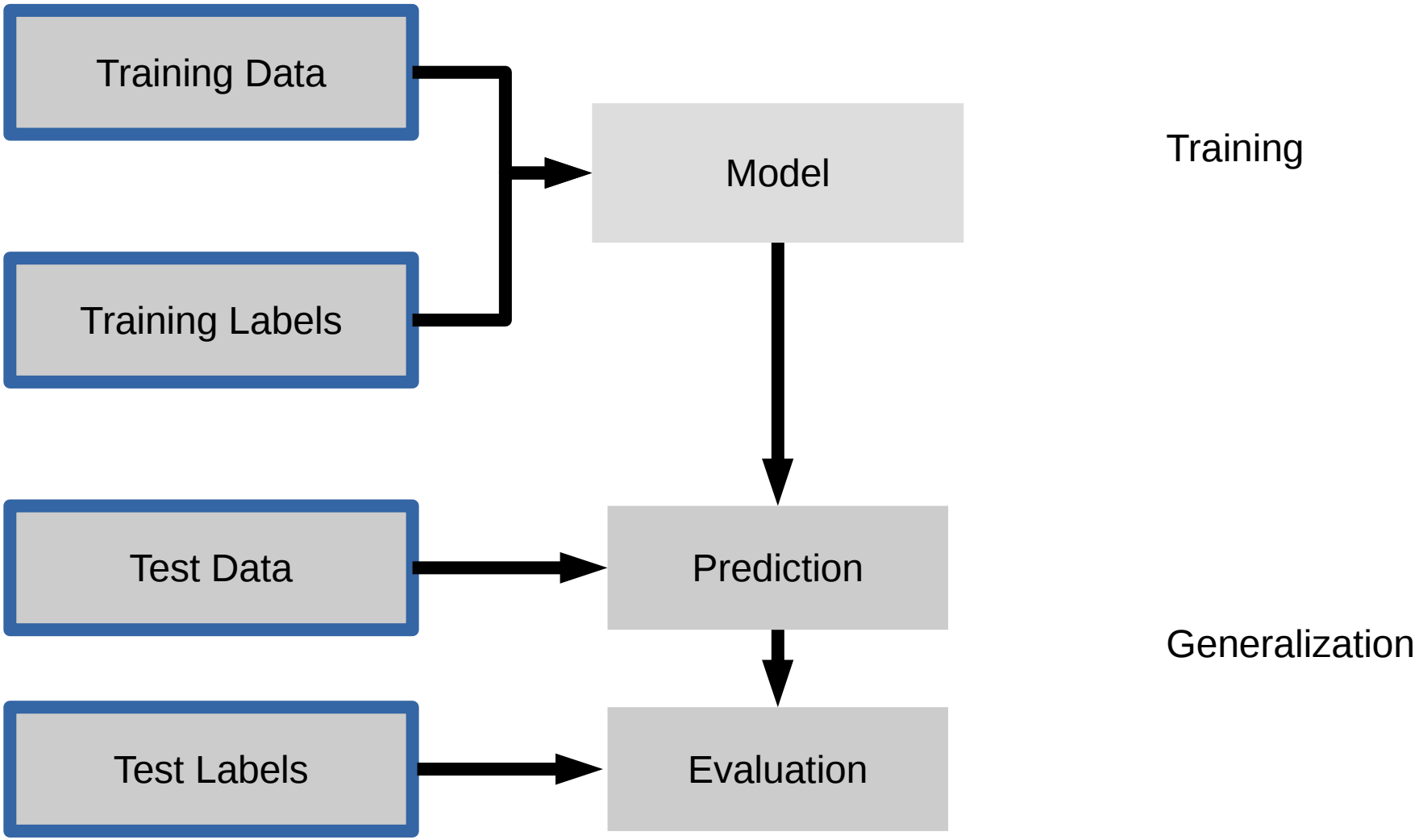
$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix}$$

one feature

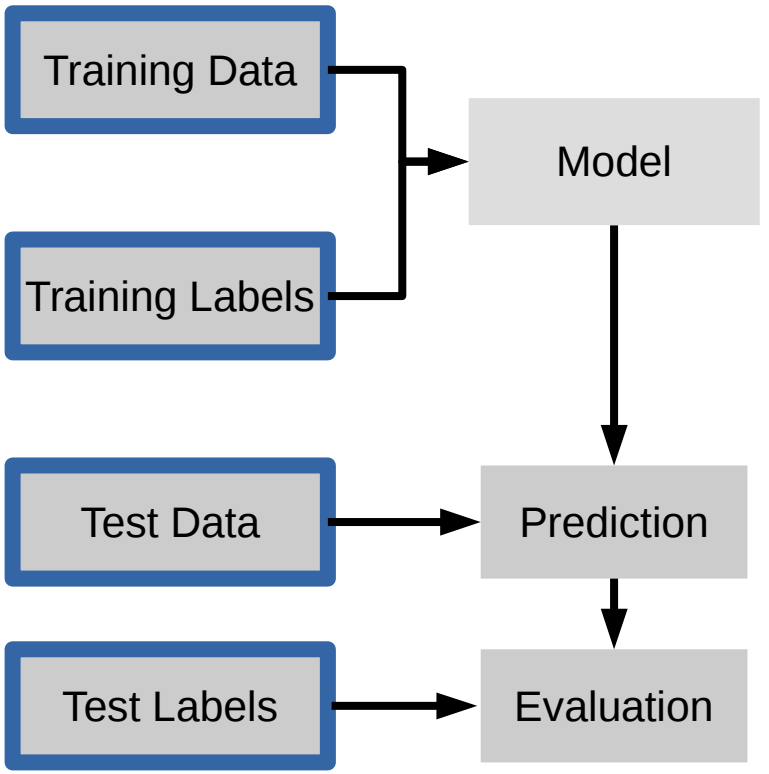
$$y = \begin{pmatrix} 1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7 \end{pmatrix}$$

outputs / labels

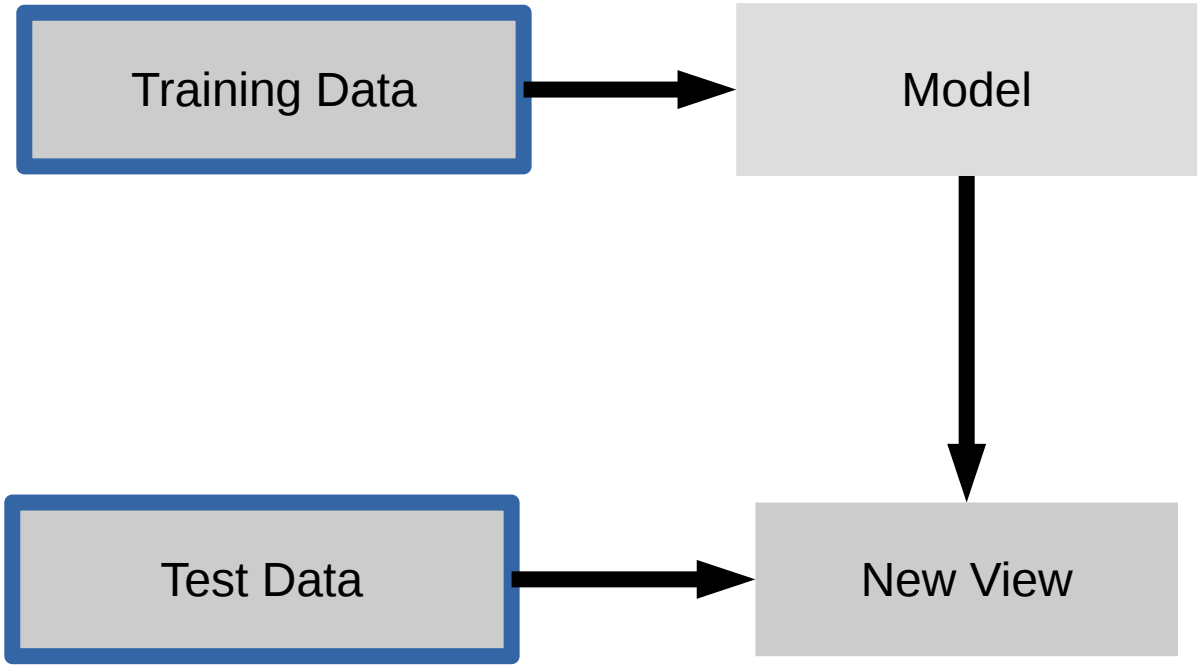
Supervised Machine Learning



```
clf = RandomForestClassifier()

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

clf.score(X_test, y_test)
```



Unsupervised Machine Learning

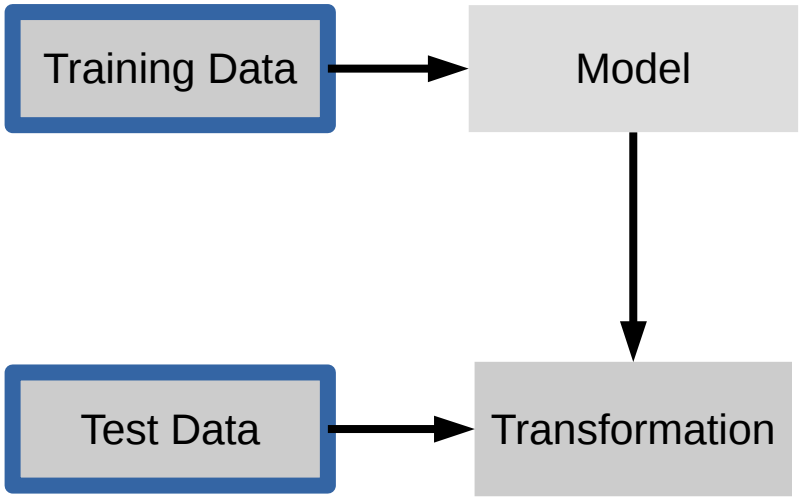


Unsupervised Transformations

```
pca = PCA()
```

```
pca.fit(X_train)
```

```
X_new = pca.transform(X_test)
```

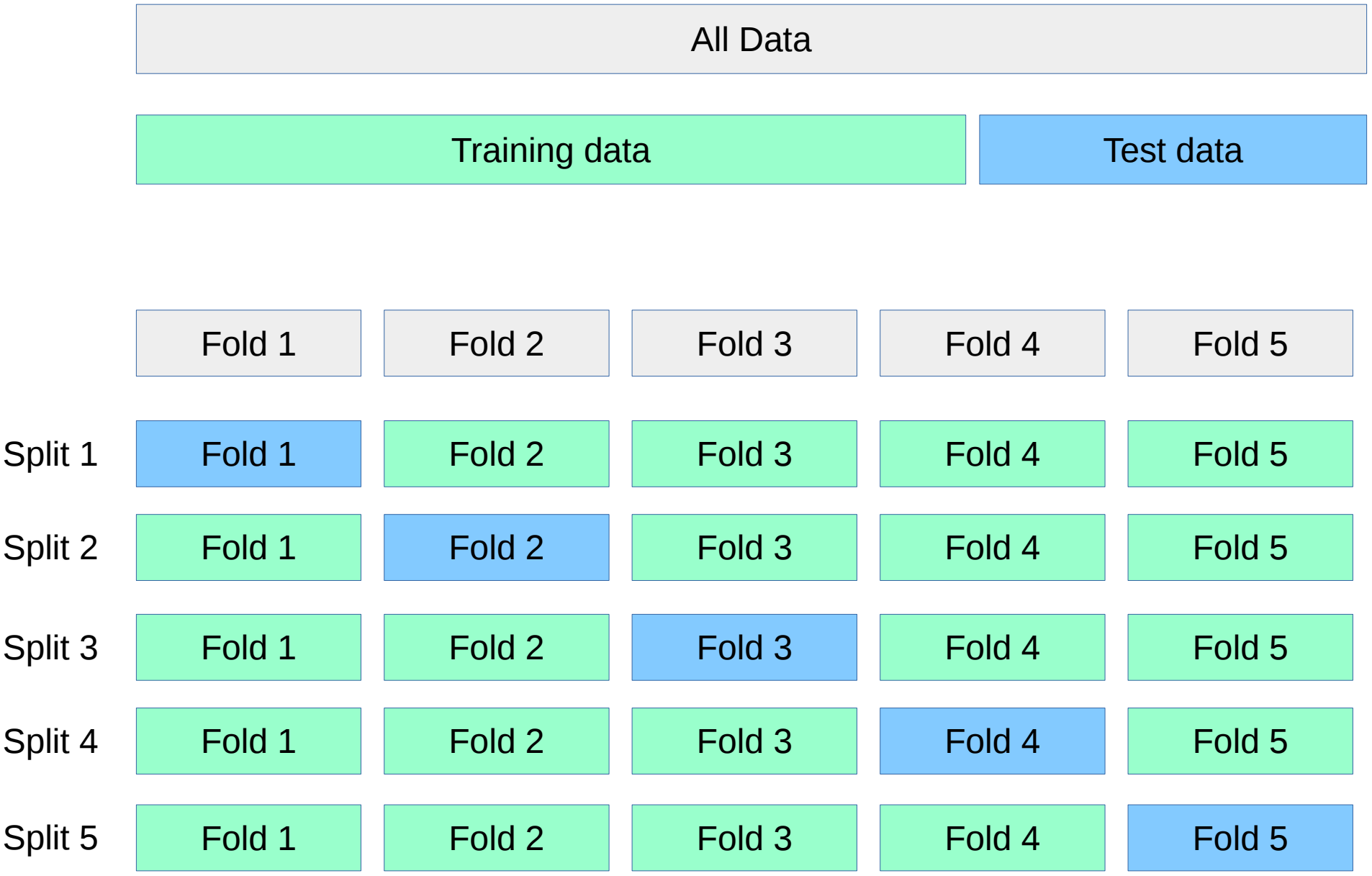


Basic API

`estimator.fit(X, [y])`

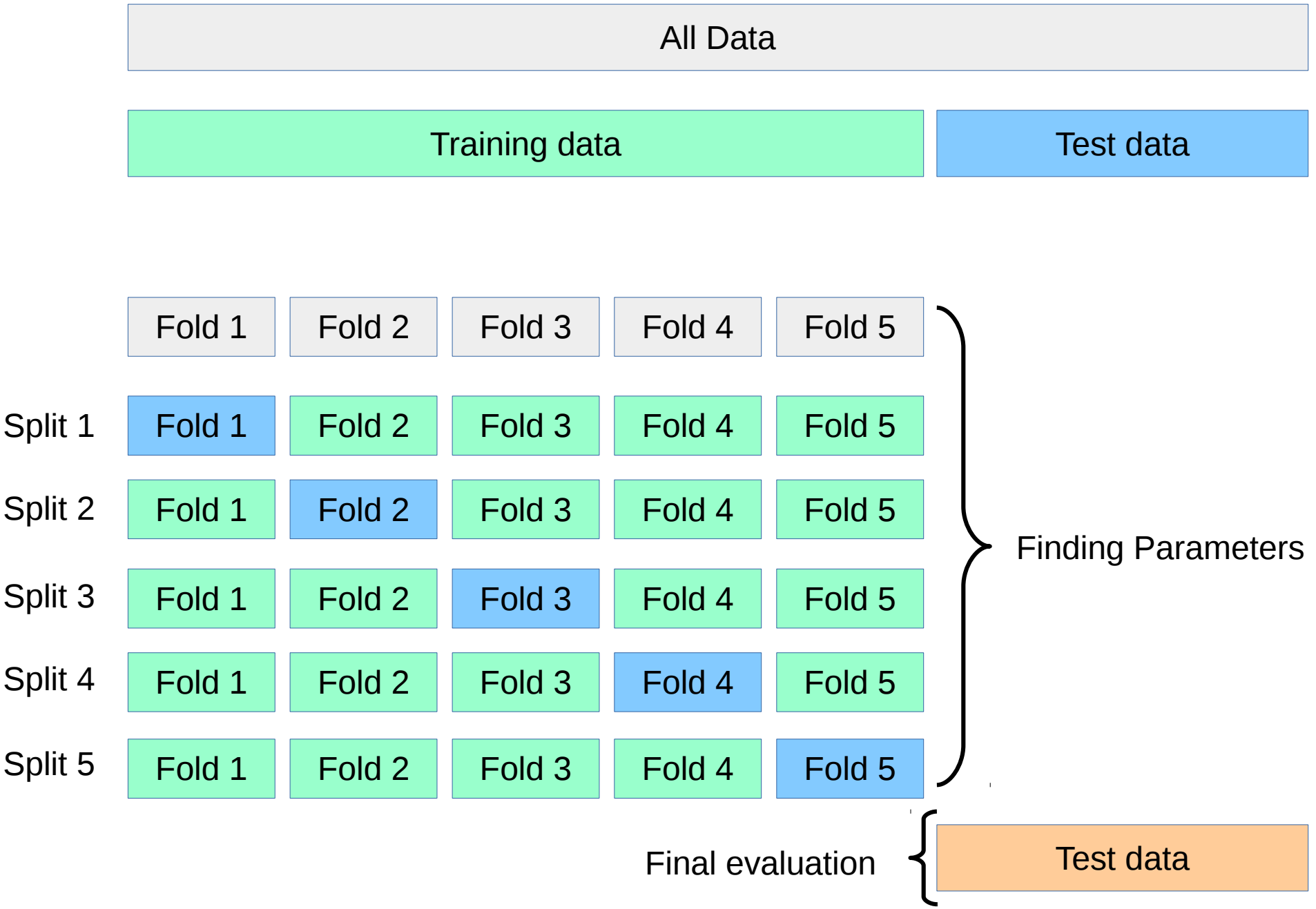
`estimator.predict`	**`estimator.transform`**
Classification	Preprocessing
Regression	Dimensionality reduction
Clustering	Feature selection
	Feature extraction

Model Evaluation and Model Selection


All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Split 2
Split 3
Split 4
Split 5

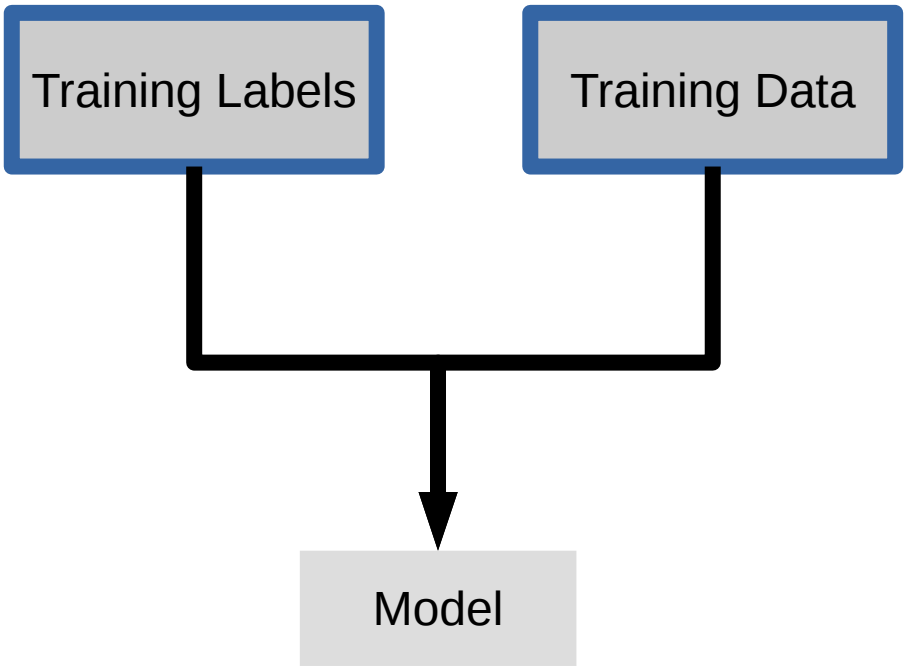
Cross-Validation

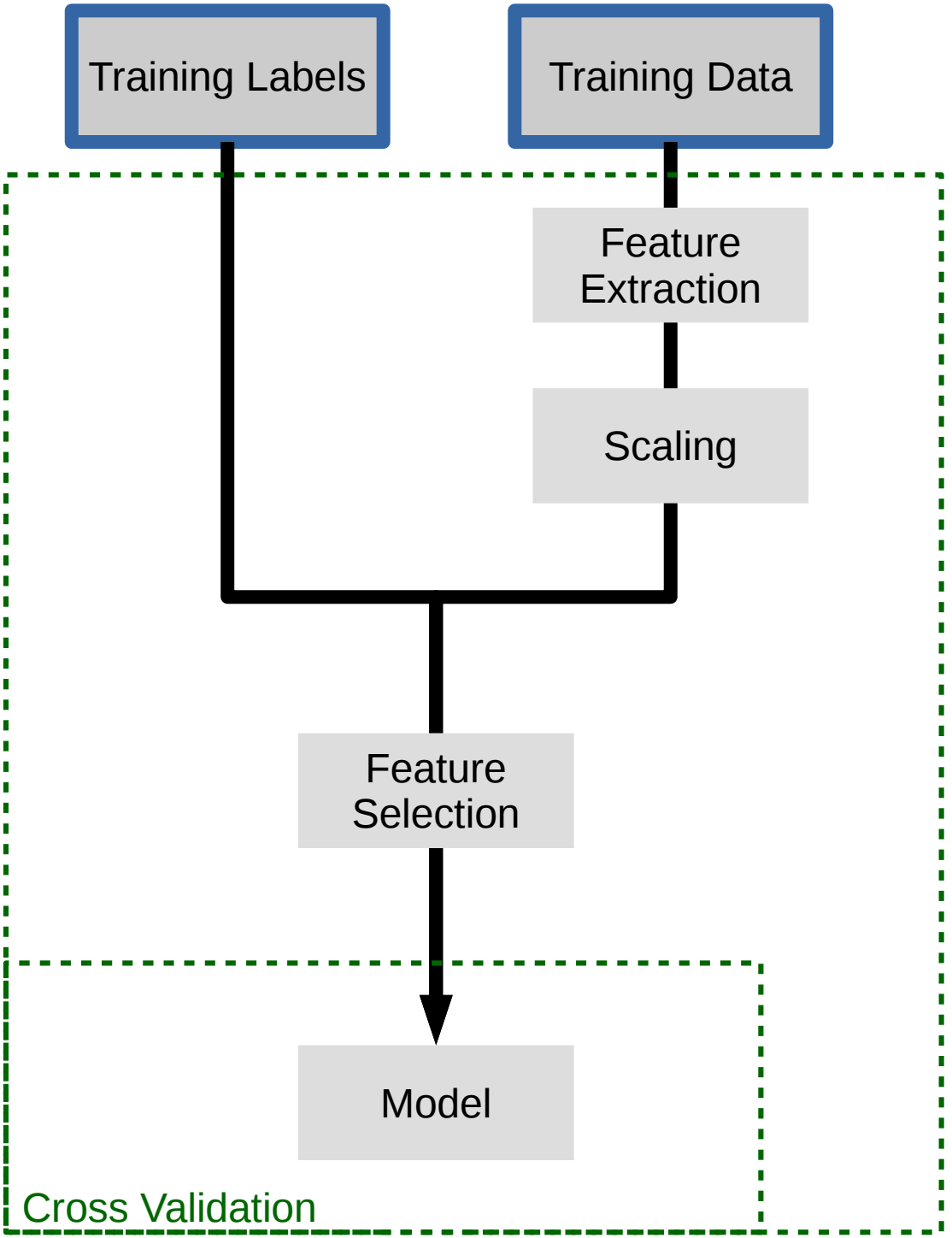
```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(SVC(), X, y, cv=5)
print(scores)

>> [ 0.92  1.    1.    1.    1.  ]
```


All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Split 2
Split 3
Split 4
Split 5
Finding Parameters
Final evaluation
Test data

Cross -Validated Grid Search

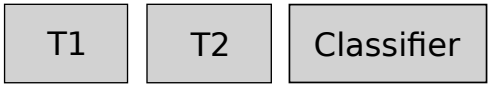
```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y)

param_grid = {'C': 10. ** np.arange(-3, 3),
              'gamma': 10. ** np.arange(-3, 3)}
grid = GridSearchCV(SVC(), param_grid=param_grid)
grid.fit(X_train, y_train)
grid.predict(X_test)
grid.score(X_test, y_test)
```


Training Labels
Training Data
Model

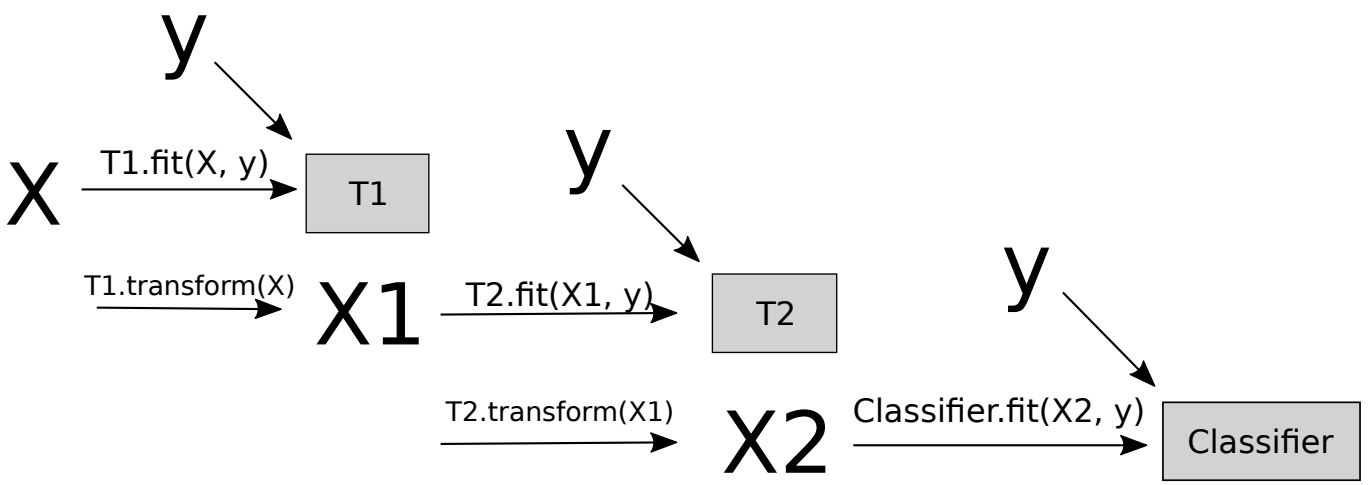

Training Labels
Training Data
Feature Extraction
Scaling
Feature Selection
Model
Cross Validation

Pipelines

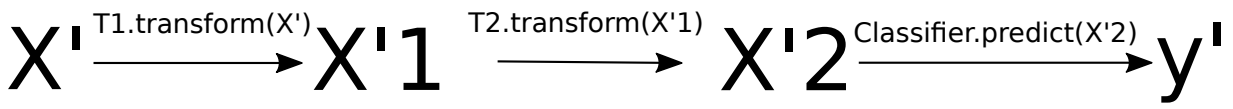
```
pipe = make_pipeline(T1(), T2(), Classifier())
```




```
pipe.fit(X, y)
```




```
pipe.predict(X')
```



Pipelines

```
from sklearn.pipeline import make_pipeline

pipe = make_pipeline(StandardScaler(), SVC())
pipe.fit(X_train, y_train)
pipe.predict(X_test)
```

```
param_grid = {'svc__C': 10. ** np.arange(-3, 3),
              'svc__gamma': 10. ** np.arange(-3, 3)}

scaler_pipe = make_pipeline(StandardScaler(), SVC())
grid = GridSearchCV(scaler_pipe, param_grid=param_grid, cv=5)
grid.fit(X_train, y_train)
```

Combining Pipelines and Grid Search

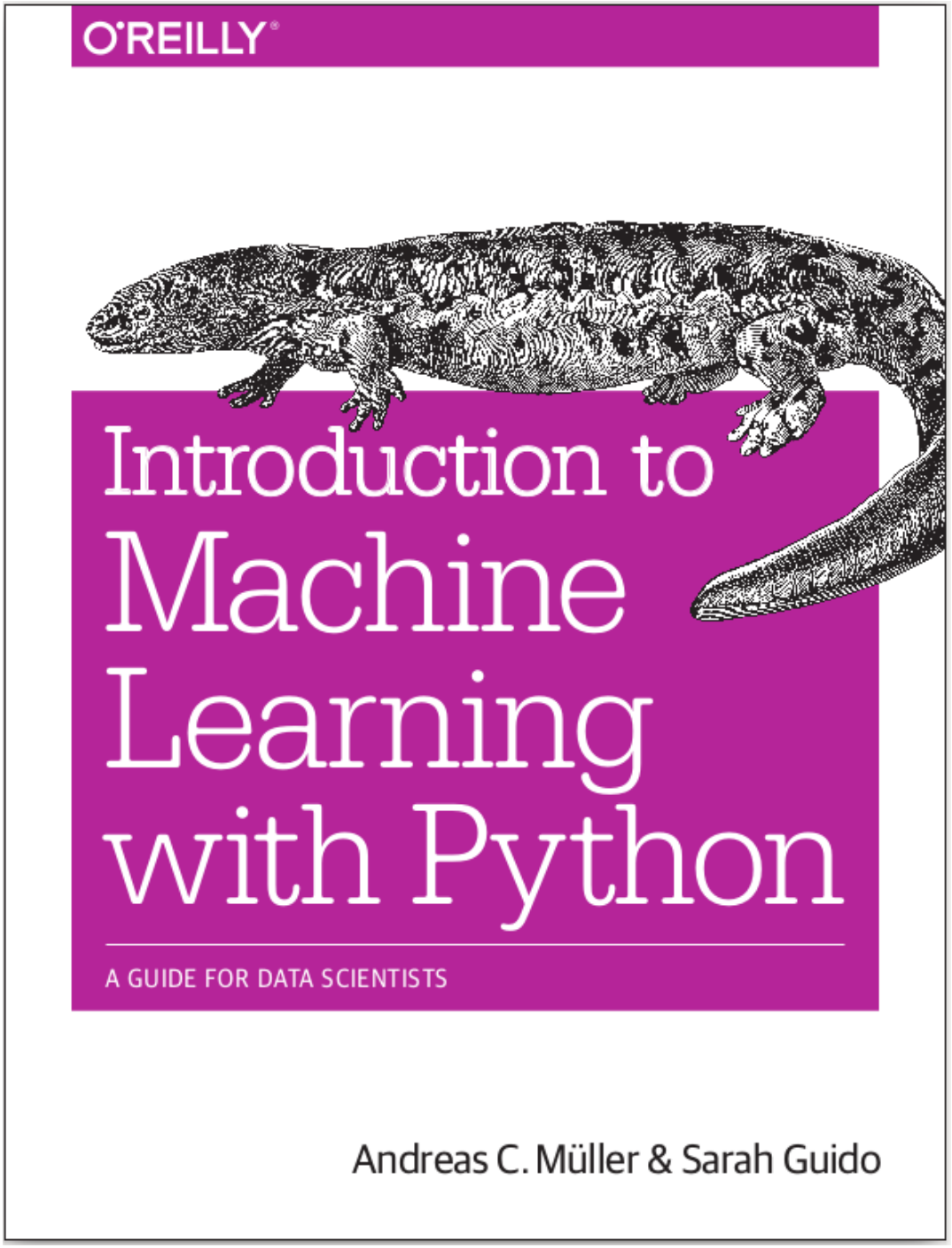
Proper cross-validation

```
param_grid = {'svc__C': 10. ** np.arange(-3, 3),
              'svc__gamma': 10. ** np.arange(-3, 3)}

scaler_pipe = make_pipeline(StandardScaler(), SVC())
grid = GridSearchCV(scaler_pipe, param_grid=param_grid, cv=5)
grid.fit(X_train, y_train)
```

Combining Pipelines and Grid Search II

Searching over parameters of the preprocessing step

```
param_grid = {'selectkbest__k': [1, 2, 3, 4],
              'svc__C': 10. ** np.arange(-3, 3),
              'svc__gamma': 10. ** np.arange(-3, 3)}

scaler_pipe = make_pipeline(SelectKBest(), SVC())
grid = GridSearchCV(scaler_pipe, param_grid=param_grid, cv=5)
grid.fit(X_train, y_train)
```



amueller.github.io

@amuellerml

@amueller

t3kcit@gmail.com